LAB 5

Objective: Use **minikube**, explore the Kubernetes API, create your first Pod, create your first namespace, and start using the **kubectl** CLI

Explore the API

Start **minikube** and **ssh** to it:

```
$ minikube start
$ minikube ssh
```

You are now logged into **minikube**, you will see that you have Docker running and a few containers are already running. Access the API server and check the various endpoints:

```
$ curl localhost:8080
$ curl localhost:8080/version
$ curl localhost:8080/api/v1
```

You will see the various API groups, and the resources that belong to each group.

Create your first Pod

Exit **minikube** to get back on a terminal in your local machine, check that you have **kubectl** running and list the Pods:

```
$ kubectl version
$ kubectl get pods
```

Now, create your first Pod using the YAML file provided in the *Course Resources* (for a refresher on how to access the *Course Resources*, look at page 1.8): **redis.yaml**

```
$ kubectl create -f redis.yaml
```

Copyright 2017. The Linux Foundation. All rights reserved.

Once the container is running, you will be able to access the logs of the container:

```
$ kubectl logs redis
```

And you will be able to *enter* the container and run the `redis` CLI:

```
$ kubectl exec -ti redis redis-cli
127.0.0.1:6379>
```

Create a Pod in a specific namespace

To avoid name collision, you can create *namespaces*. In each namespace, you can create Kubernetes resources. Let's create an `lfs248` namespace and create another `redis` pod in that namespace:

```
$ kubectl create ns lfs248
```

Check that your namespace has been created:

```
$ kubectl get ns
NAME          STATUS    AGE
default       Active    1d
kube-system   Active    1d
lfs248        Active    1m
```

You will see a `kube-system` namespace, which is created automatically by `minikube` and contains system specific services.

To create a Pod in a specific namespace, you need to specify it in the metadata of the Pod. For example, to create a `redis` Pod in the `lfs248` namespace. In the YAML file below, you see that the namespace in the metadata has changed from `default` to `lfs248`:

```
$ cat redis-lfs248.yaml
apiVersion: v1
kind: Pod
metadata:
  namespace: lfs248
  name: redis
spec:
  containers:
    - name: redis
      image: redis
```

Create the Pod and check that it is running in the correct namespace. You can then use the `logs` command by passing a `namespace` option:

```
$ kubectl create -f redis-lfs248.yaml
$ kubectl get ns
$ kubectl logs redis --namespace=lfs248
```

Explore kubectl

We just saw how to use **`kubectl`** to create Pods based on files, list pods and namespaces, access container logs and create namespaces. There is much more to **`kubectl`**, check the usage to see some of the commands available:

```
$ kubectl --help
```

Since all resources are managed via REST endpoints, you can use **`kubectl`** to delete the Pods and namespace you just created:

```
$ kubectl delete pods redis
$ kubectl delete pods redis --namespace=lfs248
$ kubectl delete ns lfs248
```

If you want to make the link with the API endpoints, check the very helpful **`kubectl --v=99`** verbose mode.

Copyright 2017. The Linux Foundation. All rights reserved.